CSC 263 Summer 2018, Assignment 1

You are allowed to work in a group of at most 3 members, and make one submission as a group. You should clearly write names of all group members in the first page of your submission. Everyone in the same group will get the same marks.
Please read and understand the policy on Academic Honesty on the course website. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes). For example, indicate clearly the **name** of every student with whom you had discussions, the **title and sections** of every textbook you consulted (including the course textbook), the **source** of every web document you used (including documents from the course webpage), etc.
For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.

1. (*10 points*) **Inversions.** Let $A[1 \ldots n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then we call the pair (i, j) an *inversion* of A.

(a) (1 point) List all inversions of the array $[2, 4, 7, 5, 1]$.

(b) (2 points) Which array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many inversions does it have?

(c) (2 point) What is the relationship between the running time of insertion sort and the number of inversions in the input array (write the running time as a function of the number of inversions M and the input size n)? Justify your answer.

(d) (5 points) Give an algorithm running in $\Theta(n \log n)$ that counts the number of inversions in an input array of n elements. Write the pseudocode; justify your algorithm is correct and the running time is $\Theta(n \log n)$.

2. (*5 points*) **Extract Second Largest.** Consider a max-priority queue Q implemented using a binary max-heap. We would like to design an ExtractSecondLargest(Q) operation, which returns the second largest key in Q and deletes it from Q. The worst-case running time of this operation must be in $O(\log n)$. We assume that all keys in Q are distinct integers.

(a) (3 points) Write the pseudocode of your ExtractSecondLargest(Q) algorithm. Let Q be an array whose indices start from 1. You can use operations from the textbook and lectures as helpers without describing their details.

(b) (2 points) Explain why your pseudocode works correctly. In particular, explain why the element you extract is the second largest one, and why this operation maintains the heap property. Also explain why the worst-case running time of your algorithm is in $O(\log n)$.

3. (*5 points*) **Heap Delete.** The operation HeapDelete(A, i) deletes the item in node i from max-heap A.

(a) (3 points) Write pseudocode for HEAPDELETE which runs in time $O(\log n)$ for an n-element max-heap.

(b) (2 points) Explain why your pseudocode works correctly, and why the running time is $O(\log n)$.

4. (*15 points*) ***d*-ary Heap.** A *d-ary heap* is like a binary heap, except that non-leaf nodes have at most d children instead of 2 children.

(a) (2 point) How would you represent a d-ary heap in a array? In particular, given a node with index i, what are the indices the its parent and children nodes?

(b) (2 points) What is the height of a d-ary heap of n elements in terms of n and d?

(c) (4 point) Write pseudocode for implementing EXTRACTMAX in a d-ary max-heap. Analyze its running time in terms of n and d (express the running time using big-O notation and justify your answer).

(d) (4 point) Write pseudocode for implementing INCREASEKEY(A, i, k) in a d-ary max-heap. Analyze its running time in terms of n and d (express the running time using big-O notation and justify your answer).

(e) (3 point) Write pseudocode for implementing INSERT in a d-ary max-heap. Analyze its running time in terms of n and d (express the running time using big-O notation and justify your answer).

5. (*20 points*) **Reverse Young Tableaus.** An $m \times n$ *reverse Young tableau* is an $m \times n$ matrix such that the entries of each row are in descending order (reversely sorted) from left to right, and the entries of each column are in descending order from top to bottom. Some of the entries of a reverse Young tableau may be $-\infty$, which are treated as nonexistent elements. Thus, a reverse Young tableau can be used to hold $r \leq mn$ finite numbers.

(a) (1 point) Draw a 4×4 reverse Young tableau containing the elements $\{9, 15, 3, 1, 4, 8, 5, 14, 12\}$.

(b) (2 points) Argue that an $m \times n$ reverse Young tableau Y is empty if $Y[1, 1] = -\infty$. Argue that Y is full if $Y[m, n] > -\infty$.

(c) (5 point) Write pseudocode for implementing EXTRACTMAX on a nonempty $m \times n$ reverse Young tableau, running in time $O(m + n)$ (justify the running time).

(d) (4 point) Write pseudocode for implementing INSERT, which adds a new element into a nonfull $m \times n$ reverse Young tableau in $O(m + n)$ time (justify the running time).

(e) (5 point) Write pseudocode for implementing SEARCH to determine whether a given number is stored in a given $m \times n$ reverse Young tableau in $O(m+n)$ time (justify the running time).

(f) (3 point) Write pseudocode of an algorithm to sort n^2 numbers in an $n \times n$ reverse Young tableau in $O(n^3)$ time, using no other sorting method as a subroutine (justify the running time).

6. (*5 points*) **Find the Median.** Write pseudocode for SEARCHMEDIAN(x) which solves the following problem in $O(n)$ time: Given a binary search tree rooted at x, return the node with the median key. As in the lecture, each node has the following attributes: key, left, right, parent. Suppose there are n nodes in the tree; the median key is the $\lceil n/2 \rceil$-th value when all the keys of the tree is fully sorted. (Note that the number of nodes is not known in advance; the only parameter passed in is the root node x.) Justify your algorithm is correct, and the running time is $O(n)$.